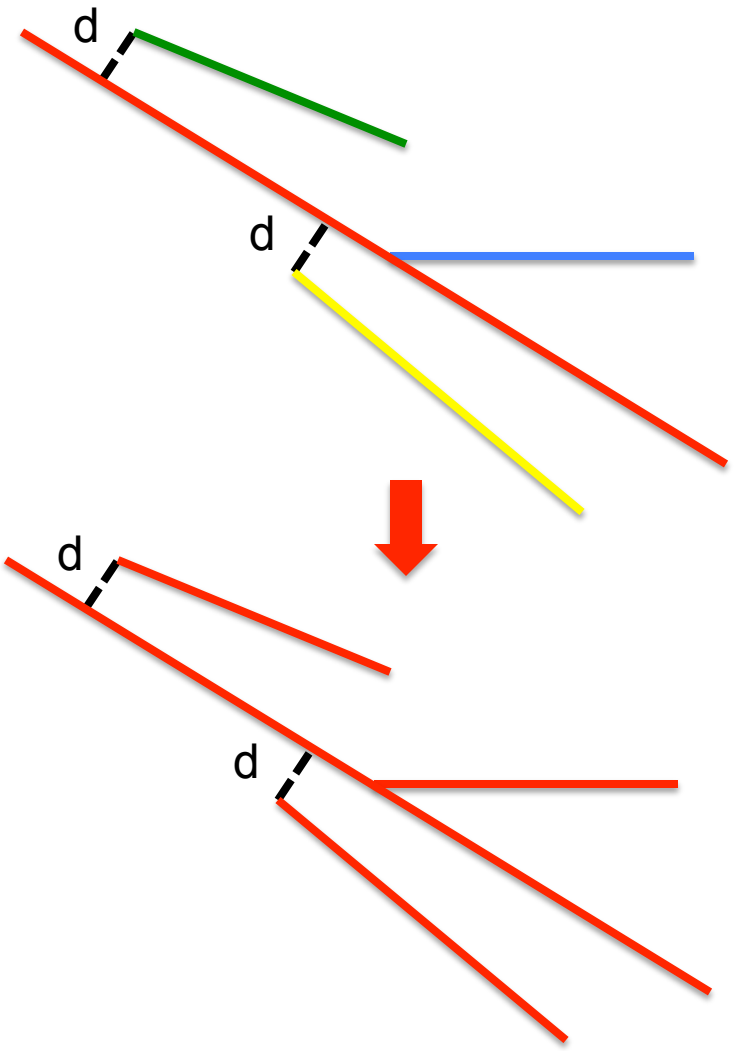
Update on fuzzy clustering

Ben Carls

Since the last update

- Concentrating on evaluations, will demonstrate efficiency and purity
- A few things have changed, a lot of effort has gone into examining showers
  - Difficult to reconstruct, difficult to identify versus tracks
  - Now taking advantage of tools developed by Andrzej Szelc

Handling showers



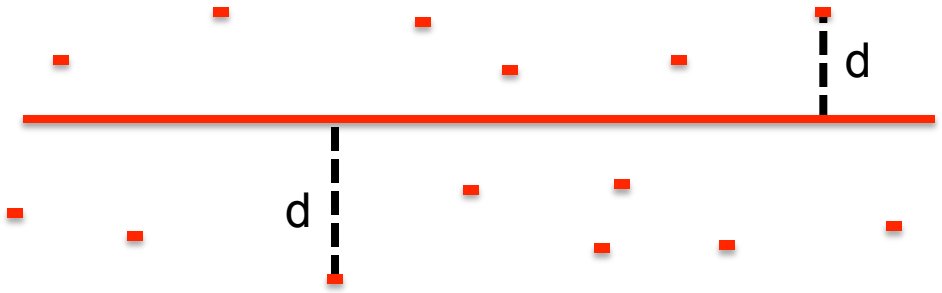
Hough line finder very effective at finding lines in showers

Using A. Szelc's figure of merit for shower identification

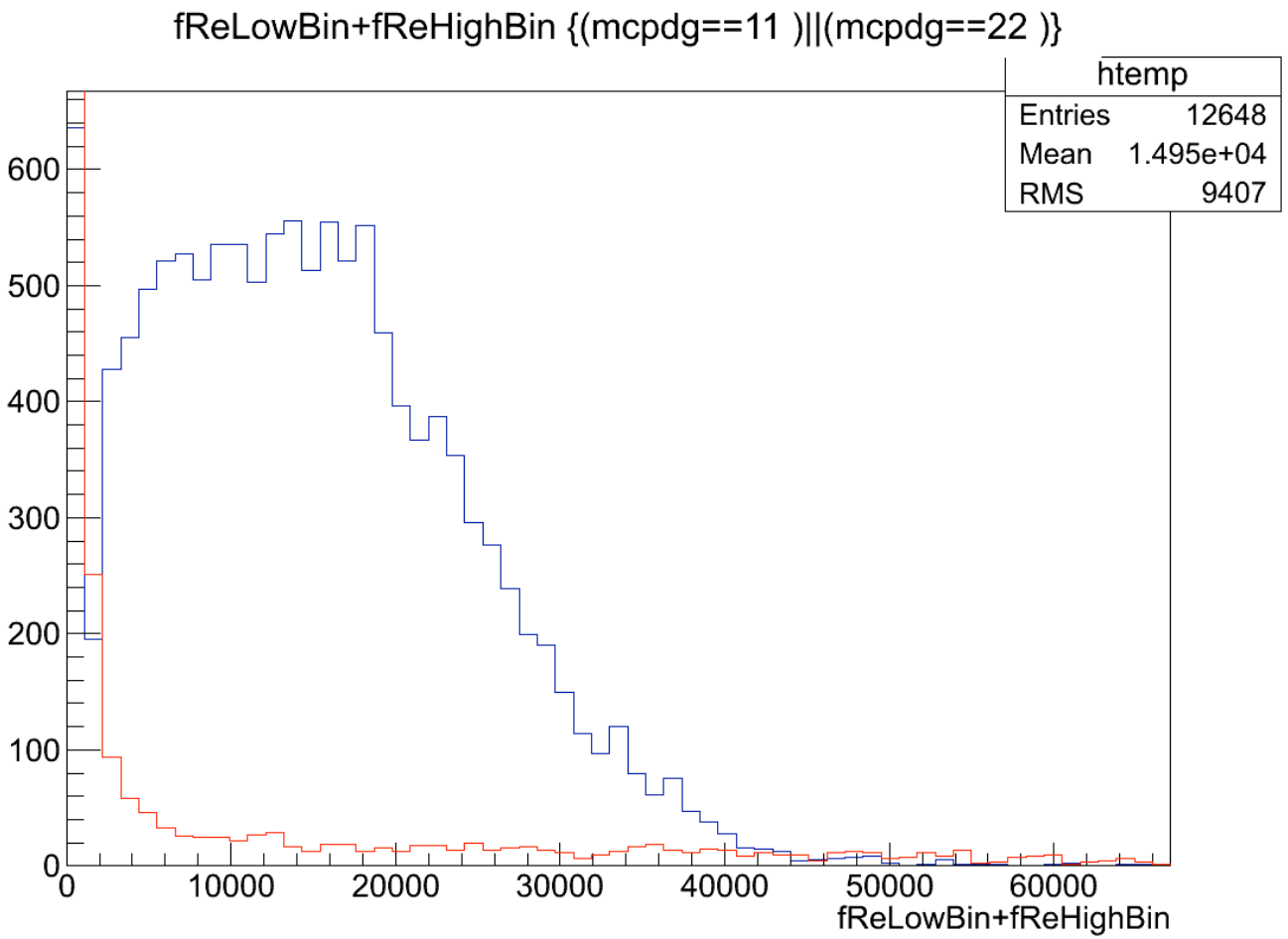
Assuming a cut on the figure of merit is met, merge lines assuming they are close enough (no angle cut)

Shower likeness

- Around each line, sum up distance from a hit to the line, then divide by the charge of the hit, value will be large for showers

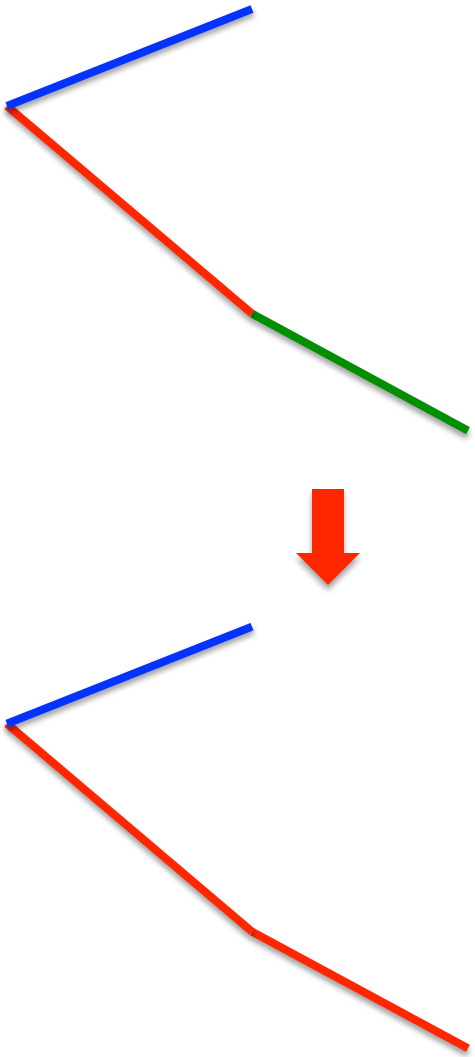


Shower likeness



ortdist = Get2DDistance(wire_on_line,time_on_line,wire,time);
reweight = (ortdist<1.) ? 0.1 : ortdist;

Tracks



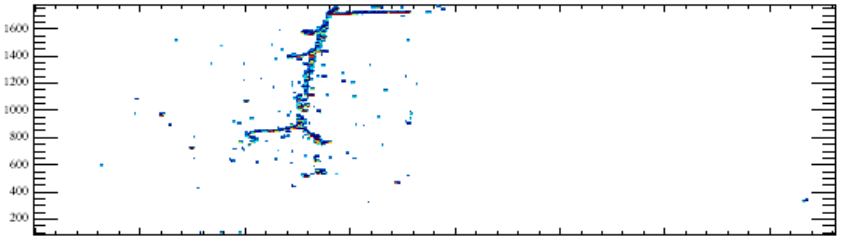
Merging tracks with slopes within 5° of each other

A second merging is then performed using a wider angle (30°), and examining charge

The average charge deposition of the 5 nearest hits of each line are used to construct an asymmetry:

$$q_{asymm} = \frac{\langle q_{line1} \rangle - \langle q_{line2} \rangle}{\langle q_{line1} \rangle + \langle q_{line2} \rangle}$$

Overview of the algorithm



Run fuzzy clustering, merge as needed

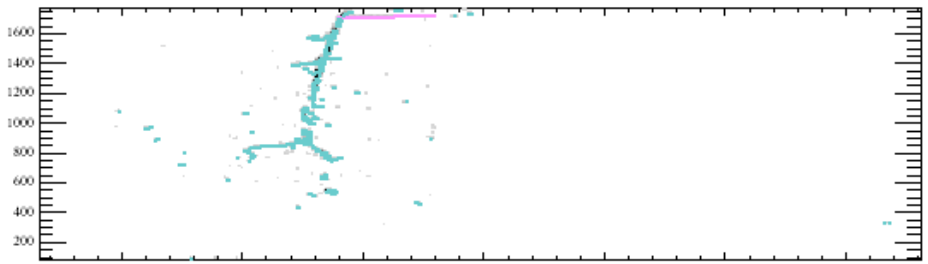
Run Hough line finder

Look and construct proto-showers

Merge tracks with slopes within 5°

Merge tracks with slopes within 30° and having similar charge at intercept

Merge fuzzy cluster remnants into merged Hough lines



Purity and Efficiency

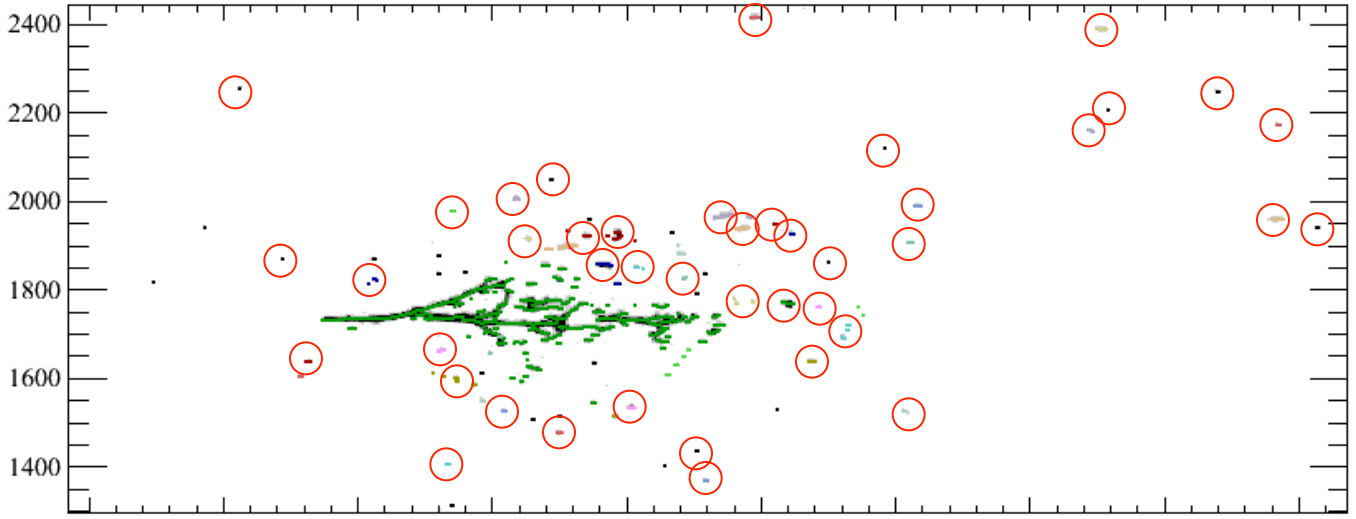
- Evaluated using the product of efficiency and purity found in the BackTracker

$$\text{Purity} = \frac{\#\,\text{hits from trackID in cluster}}{\text{total \# hits in cluster examined}}$$

$$\text{Efficiency} = \frac{\#\,\text{hits from trackID in cluster}}{\text{total \# hits for that trackID}}$$

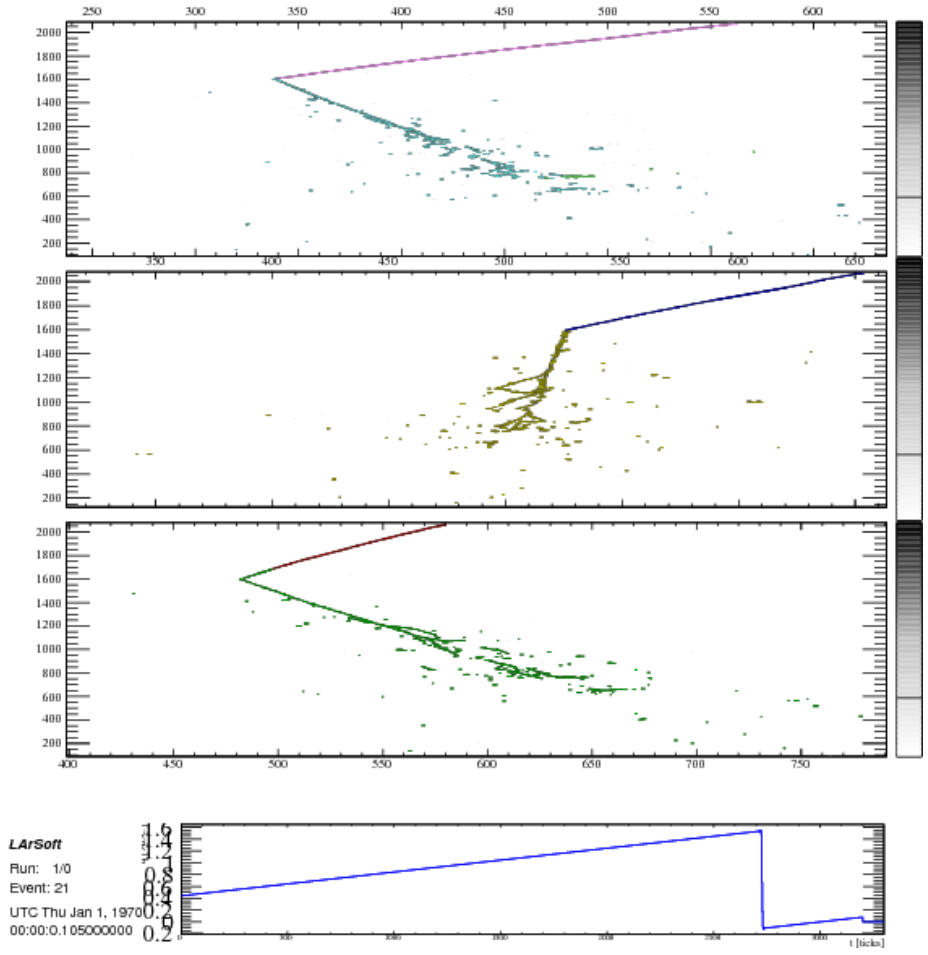
  Select values for trackIDs having the highest purity
- Comparison is tricky since Fuzzy Clustering and DBSCAN/Hough/Line Merger give different products
- Looked at 1,000 CCQE events

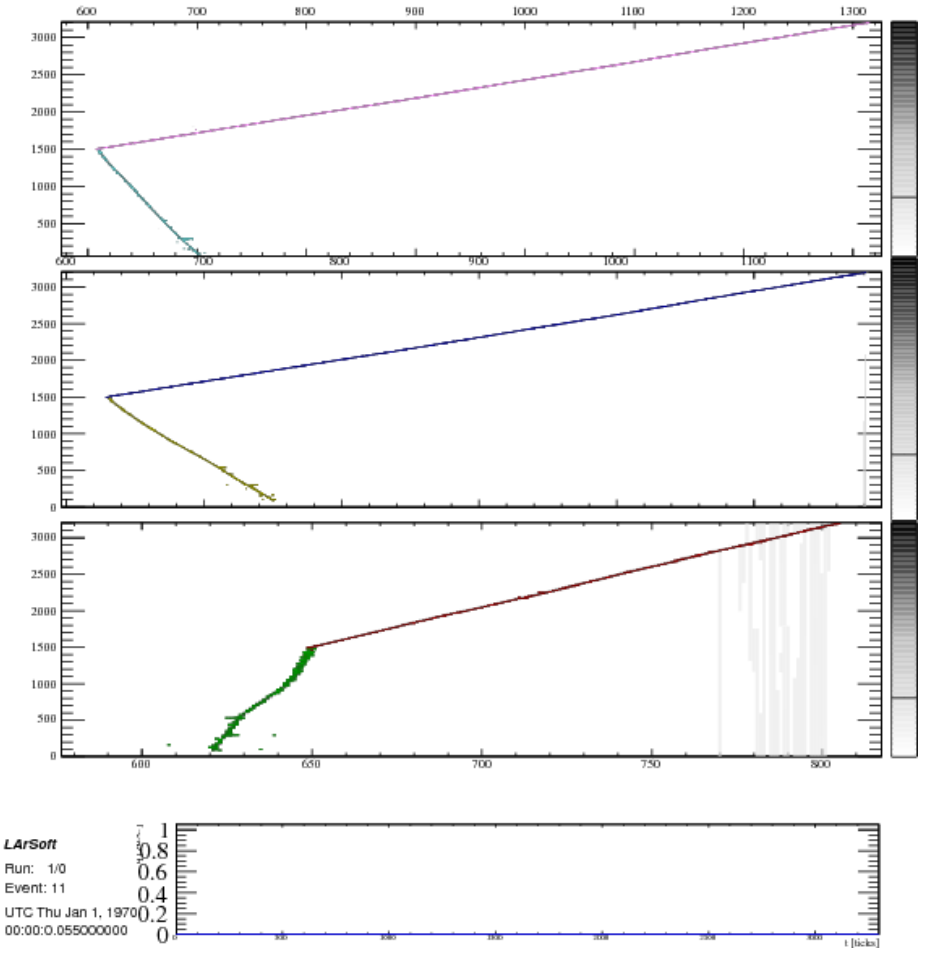
An example with DBSCAN



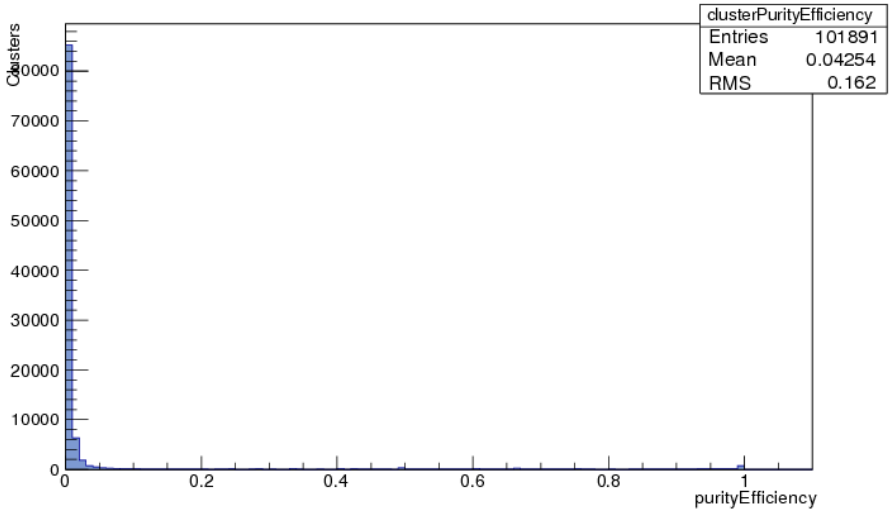
- We expect a lot of high purity clusters, lots of little clusters with only the hits from the electron shower
- We expect a lot of low efficiency clusters though, lots of the little clusters will have only a small portion of hits from the electron shower
- I've highlighted several of these in red

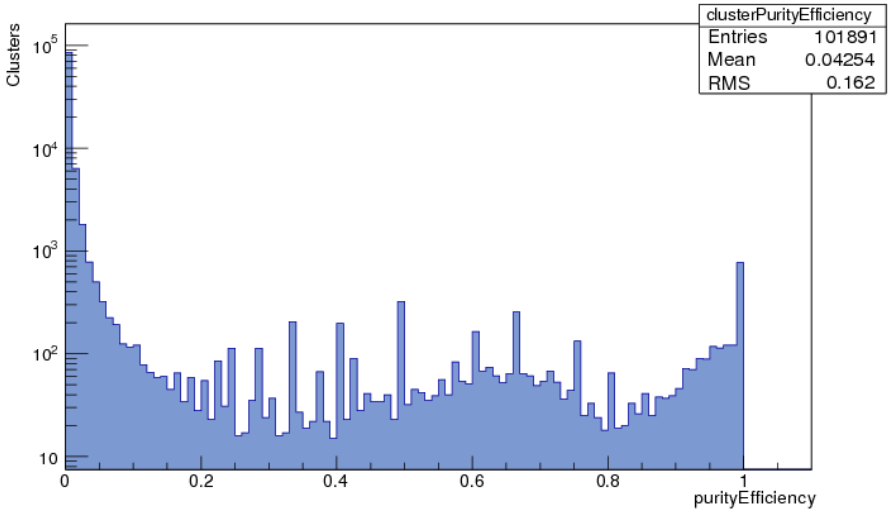
Fuzzy clustering



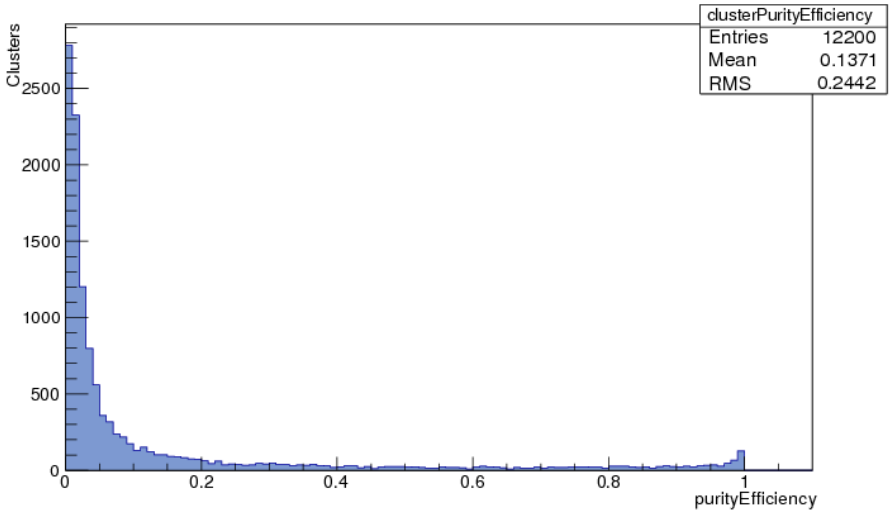


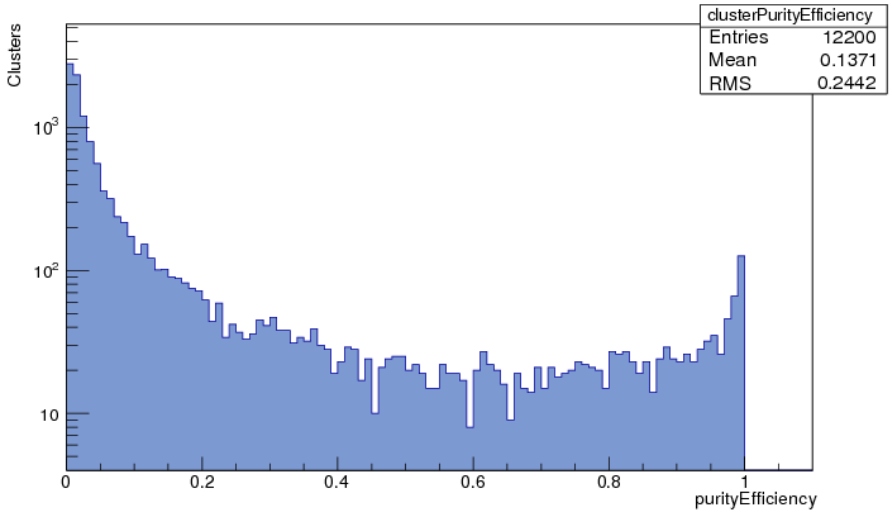
DBSCAN


clusterPurityEfficiency
Entries 101891
Mean 0.04254
RMS 0.162
Clusters
80000
70000
60000
50000
40000
30000
20000
10000
0
0
0.2
0.4
0.6
0.8
1
purityEfficiency


clusterPurityEfficiency
Entries 101891
Mean 0.04254
RMS 0.162
Clusters
10^5
10^4
10^3
10^2
10
0
0.2
0.4
0.6
0.8
1
purityEfficiency

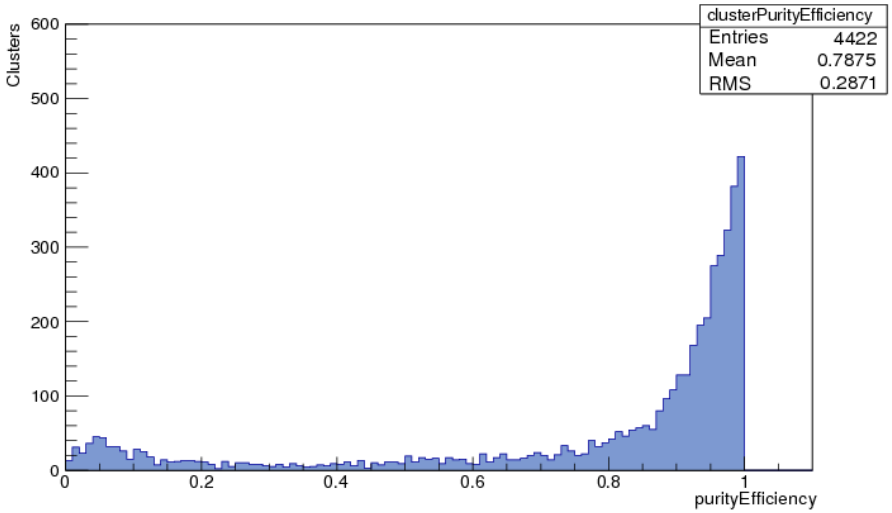
Line Merger

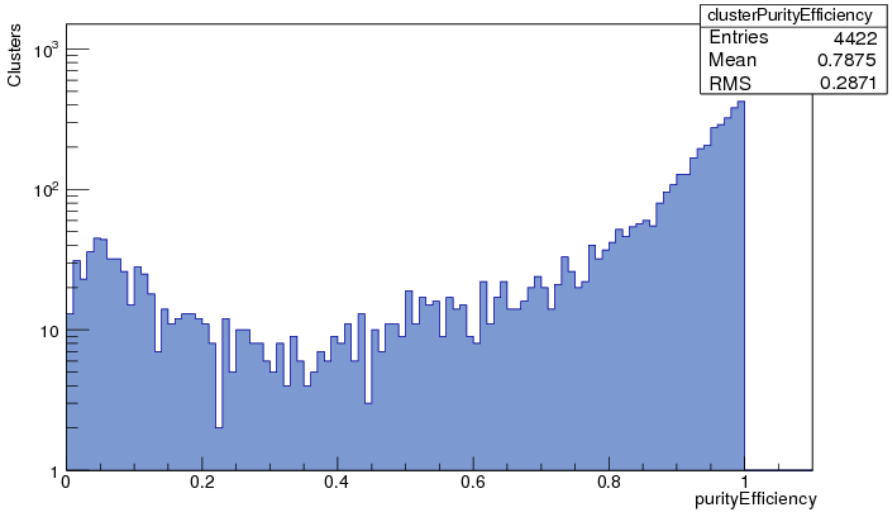




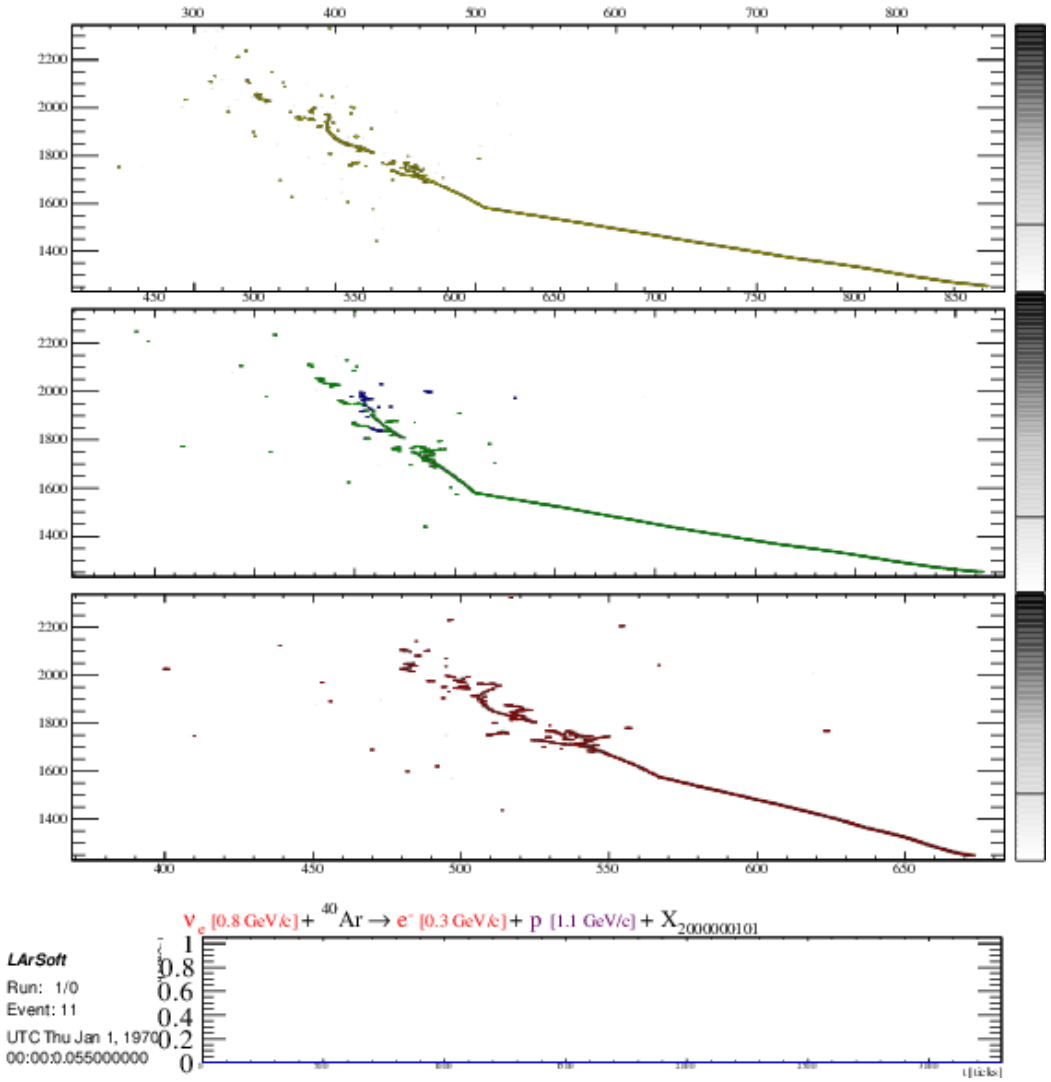
This comes from running DBSCAN, Hough Line Finder, then Line Merger

Fuzzy clustering





Shower direction issues



Algorithm got the shower direction wrong

Results in the proton being merged with the electron shower

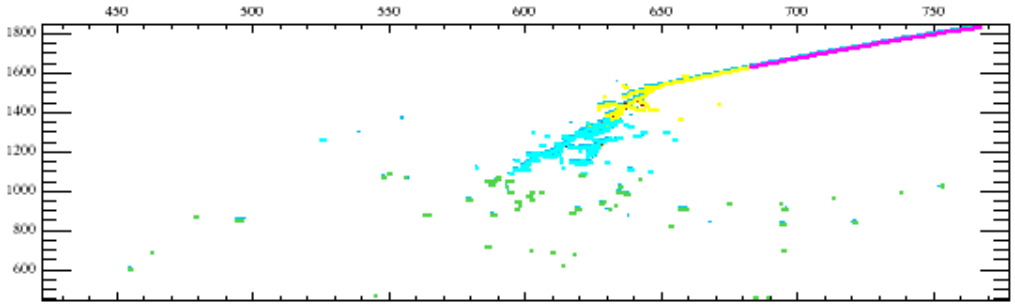
Need a better way

Where it stands now

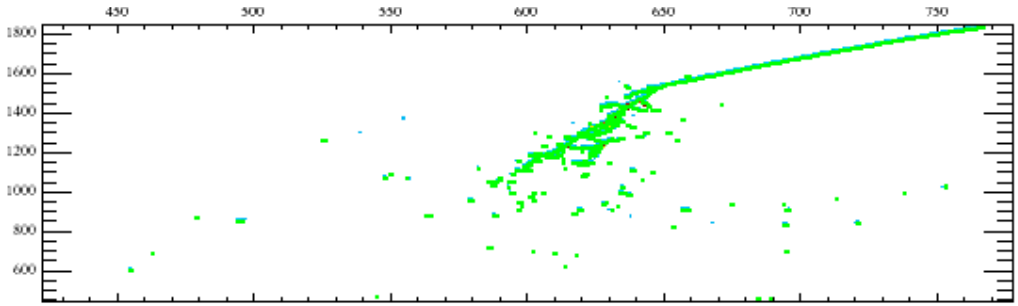
- I want to continue looking at how different parameters alter purity and efficiency
- Update documentation, draft is on Redmine
- Use new tools as they become available, specifically from Andrzej and Wes

Back up

Start with fuzzy clustering



Merge the clusters



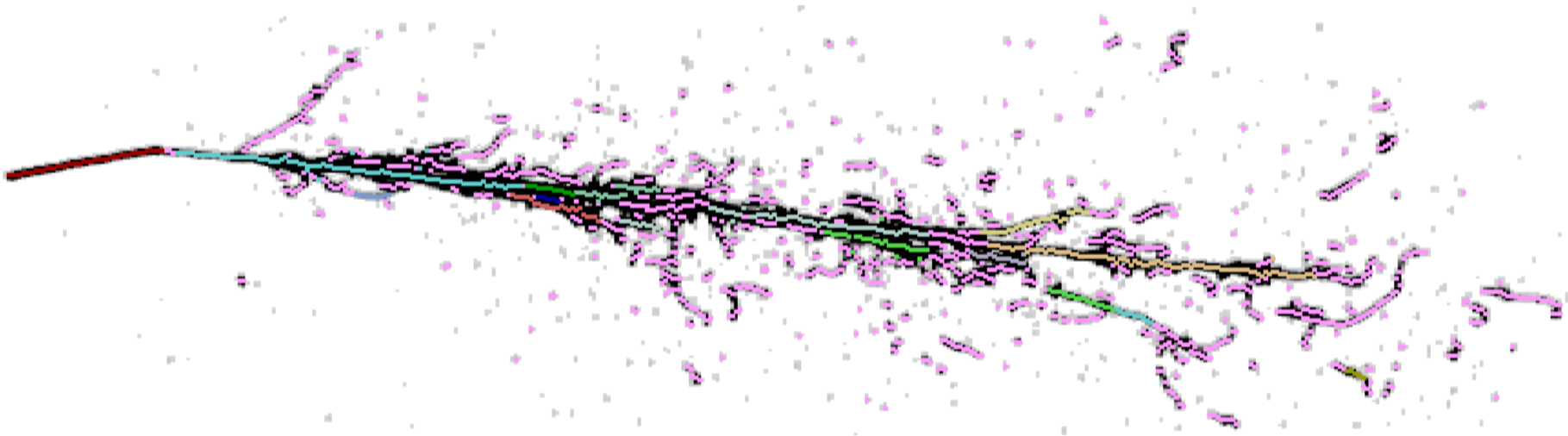
Fuzzy clustering assigns degrees of belonging, instead of definite belonging

Number of clusters determined using an optimization criteria (Xie-Beni index)

After running, the clusters are merged together

Hough line finder

- Run the Hough line finder to identify tracks and showers



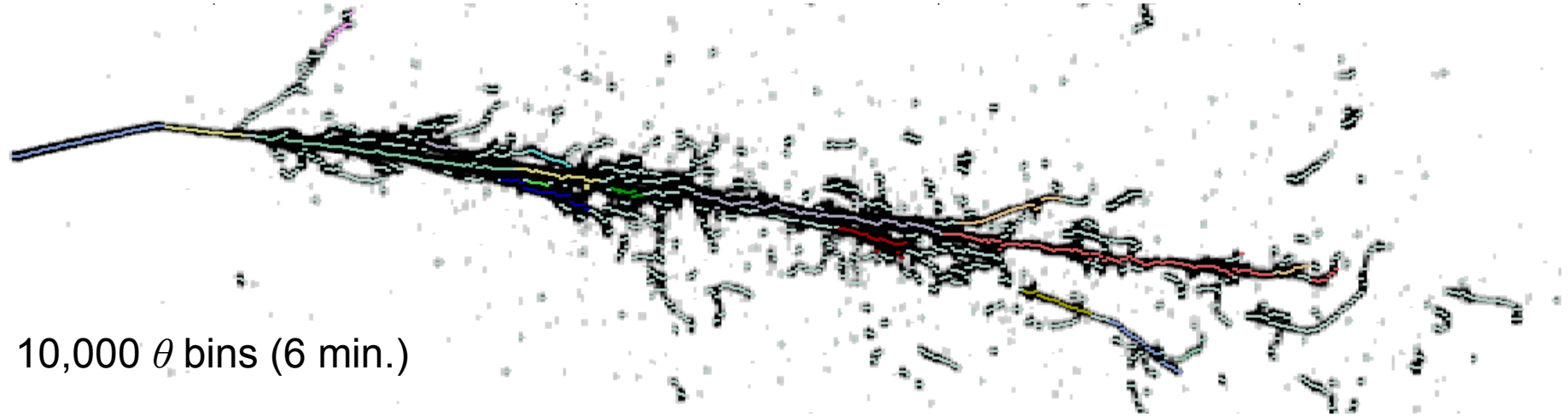
Going even faster with a PPHT

- Early work using a Progressive Probabilistic Hough Transform (PPHT) has further increased speed
- The algorithm is fairly simple:

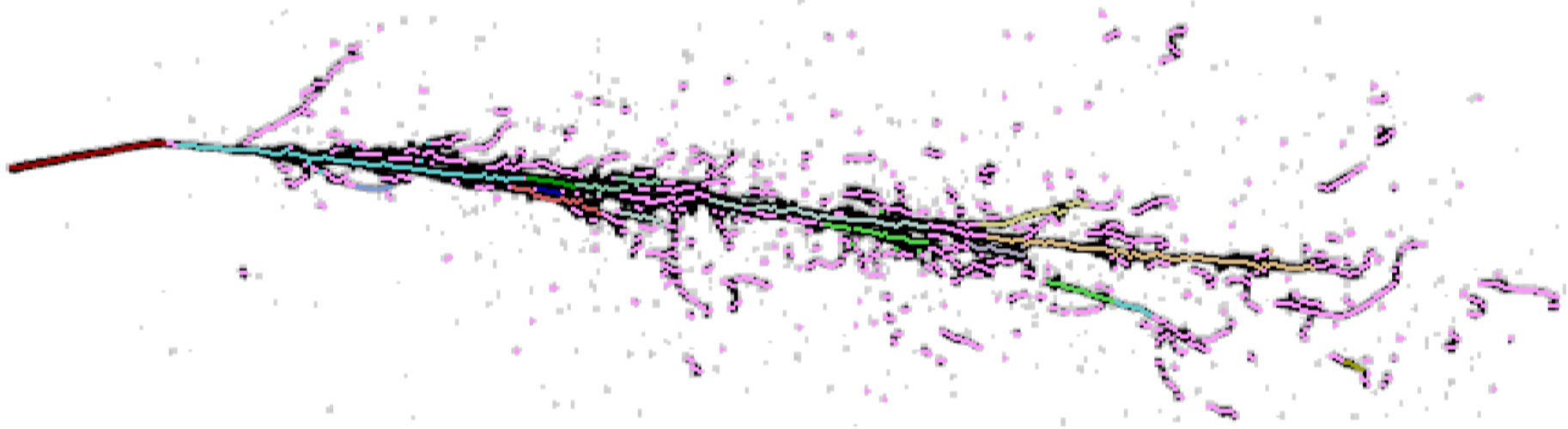
1. Randomly select one hit, remove it from the image
2. Check if the highest peak in the accumulator modified by the pixel is higher than a threshold cut; if not, go to step 1
3. Add points along the line, removing them from the image
4. Remove hits from the accumulator from the line that was found
5. Repeat step 1 if the image is not yet empty

J. Matas et al., Robust Detection of Lines Using the Progressive Probabilistic Hough Transform, Computer Vision and Image Understanding, Volume 78, Issue 1, April 2000, Pages 119–137

PPHT performance



10,000 θ bins (6 min.)



20,000 θ bins (1 min.)

Hough transform

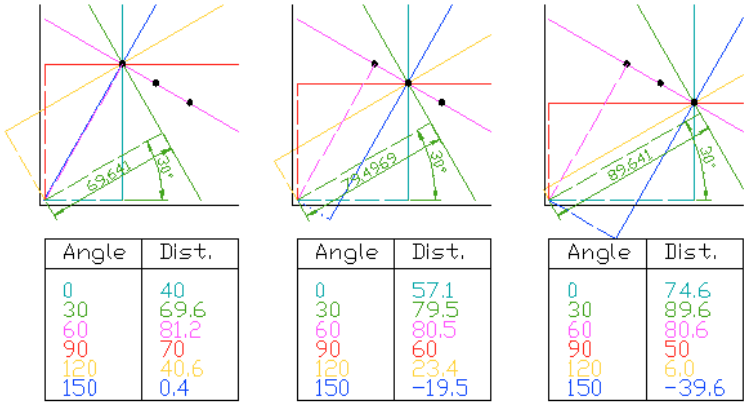
Use the parameterization:

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$$

Fill a matrix (accumulator) in (r,θ) space with:

$$r(\theta) = x_0 \cdot \cos\theta + y_0 \cdot \sin\theta$$

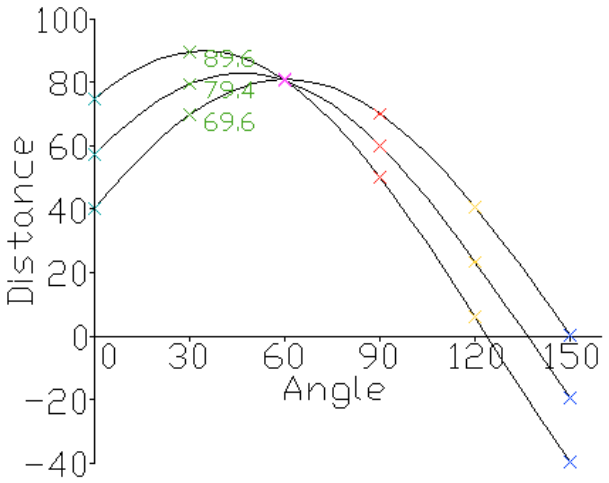
We then search the accumulator for the most populated bin, rather computationally demanding



Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5

Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6



The accumulator, binned